NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

63-3-4

EXMONT 8-2711
UPTON 0-7111

# HUGHES AIRCRAFT COMPANY

CULVER CITY
CALIFORNIA

IN REPLY REFER TO:
63H-1910/9973

10 May 1963

SUBJECT: Contract AF 04(695)-210

THRU:
Air Force Plant Representative
Hughes Aircraft Company
ATTN: RWRAA
Culver City, California

TO:
Headquarters
Space System Division
Air Force Systems Command
Air Force Unit Post Office
Los Angeles 45, California

ATTENTION: Mr. T. Caruso/SSTK

REFERENCE: Monthly Progress Report No. 6

1. Transmitted herewith are two (2) copies of the referenced report.

2. This report is submitted in compliance with Exhibit "A", paragraph VI "Reports", of the subject contract.

HUGHES AIRCRAFT COMPANY (GC)

J. Popkin
Contract Administrator
Guidance and Controls Division
Aerospace Group

JP:rw

Enclosures - 2

cc: SSTRS - at SSD - 3 copies
ASTIA - 1 copy

Attachment "A" - Techniques for Improving Computer Reliability By Logical Design (Self-Reorganizing Machines) by Ira Terris

$ 3.60

VJ

(4) $3.60
(5) 418 756
(6) NA
(7) NA
(8) NA

1

(9) Monthly Progress Report. No 6,

(11) 29 April 1963,

(12) [34]p

(13-14) NA

To

Headquarters
Space Systems Division
Air Force Systems Command
Air Force Unit Post Office
Los Angeles 45, California
ATTENTION: SSTK

(15) Contract AF04 695 210
(16-19) NA
(20) u
(21) NA

In Accordance With
Exhibit "A" to Contract AF04(695)-210
Item VI, Reports

for the Reporting Period

1-~~April~~ 28 April 1963

Project Engineer (10) Ty I. Terris
I. TERRIS

Project Manager T J Burns
T J BURNS

Contract Administrator J Popkin
J POPKIN

Systems Laboratory
Guidance and Controls Division
Aerospace Group
Hughes Aircraft Company
Culver City, California

## TABLE OF CONTENTS

## 1.0 Technical Discussion

### 1.1 Introduction

The effort in April 1963, was in the areas of:

1. Performance Valuation Procedure, and
2. Reliability Design for Computer Organizations.

### 1.2 Performance Valuation Procedure

In progress is the analysis of degraded mode errors in midcourse trajectory determination.

The basic theory applied is described in the literature as a stable computational method and expected to be insensitive to round-off errors. Direct application of the theorems published would lead to the conclusion that computation errors are negligible against instrument anomalies. Considerable time was spent to modify the existing results by introducing computation noise into the basic dynamical model. Convergence of the estimation procedure and equilibrium was found to be significantly dependent on the computation methods used and on the accuracy limitations in the digital computer.

Two error sources were found to be dominant:

1. Error in the state transition matrix, and
2. Error in the weighting factor (optimum filter).

The errors are generated during integration and matrix inversion.

In progress is the statistical treatment of round-off errors in matrix inversion. Worst case estimates are known, but seem not useful in real time applications. It is desired to find the variance rather than a worst case bound because of the probabilistic measure of performance degradation in navigation systems (ellipsoid of concentration).

The actual objective of the performance measure development, to relate computer limitations with system performance in midcourse navigation, seems almost accomplished. However, considerable time will be necessary to complete and to review the model in all details and to work out numerical examples.

### 1.3 Reliability Design for Computer Organizations

A serial, general-purpose machine has been analyzed to determine the effect of self-reorganization on machine reliability (MTBF). The purpose of this analysis was to determine if the gain from self-reorganization techniques is positive or have the components required to implement the self-reorganization methods negated the entire advantage.

In this study which is attached, the mean time between catastrophic failures is 2.3 times the MTBF for an equivalent computer of standard organization. The penalty in total machine size was an increase in component count of 18%. Since the memory was excluded, as it was covered in a previous report (SRS-622) and the memory is generally about half the computer, the increase in size for the entire computer is about 10%.

### 1.4 Follow-On Effort

Work in the above two areas will continue. In addition, work in the area of reliability programming will commence.

### 1.5 Action Items

None.

## 2.0 Trips and Visits

On 29 April 1963, Messrs T J Burns and I Terris of Hughes visited Capt J N VanDusen of SSD, and Mr J Y Miyamoto of Aerospace Corp, to discuss the progress of the project.

### 3.3 Assigned Personnel

Anton Holick, Staff Engineer

Ira Terris, Project Engineer

### 3.4 Additional Information

The aforementioned actual figures were recapitulated from Hughes Aircraft Company Tab Run dated 28 April 1963.

### 3.5 Milestone Chart

The Milestone Chart can be found on the last page of this report. It should be noted that the estimated completion dates for Milestones B and C have been moved from the end of May to the end of July. The moving of Milestone B was necessitated by the nonexistence of a complete error analysis of the effects of digital computer noise on midcourse guidance. This analysis presently is being performed. The moving of Milestone B coincides with the earlier starting of the effort for Milestone C, which was not scheduled to start until the end of May, but is being started at the beginning of May. This was necessitated by the intimate relationship of Milestones B and C which necessitates that to some extent they be performed simultaneously.

Section 3.5 MILESTONES

63H-1910/9973
Progress Report #6
29 April 1963
Page 6 of 6

COMPUTER RELIABILITY STUDY

| | 1962 | | | | | | | | 1963 | | | | | | | | | | | | 1964 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MILESTONES | 1ST | 2ND | J | A | S | O | N | D | J | F | M | A | M | J | J | A | S | O | N | D | J | F | M | A | M | J |
| A INITIAL INVESTIGATION | | | | | | \|△ | | | | | | | | | | | | | | | | | | | | |
| B PERFORMANCE VALUATION PROCEDURE | | | | | | | | | | | | | | | \|△ | | | | | | | | | | | |
| C RELIABILITY DESIGN | | | | | | | | | | | | | | | \|△ | | | | | | | | | | | |
| D RELIABILITY PROGRAMMING | | | | | | | | | | | | | | | | \|△ | | | | | | | | | | |
| E RECOMMENDED FOLLOW-ON EFFORT | | | | | | | | | | | | | | | | \|△ | | | | | | | | | | |
| F MONTHLY PROGRESS REPORTS | | | | | | | ▲ | ▲ | ▲ | ▲ | ▲ | ▲ | \|△ | \|△ | \|△ | \|△ | | | | | | | | | | |
| G FINANCIAL MANAGEMENT REPORTS (DD1097) | | | | | | | | | \|▲ | | | \|▲ | | | \|△ | | | | | | | | | | | |
| H FINAL TECHNICAL DOCUMENTARY REPORT | | | | | | | | | | | | | | | | | \|△ | | | | | | | | | |
| I | | | | | | | | | | | | | | | | | | | | | | | | | | |
| J | | | | | | | | | | | | | | | | | | | | | | | | | | |
| K | | | | | | | | | | | | | | | | | | | | | | | | | | |
| L | | | | | | | | | | | | | | | | | | | | | | | | | | |
| M | | | | | | | | | | | | | | | | | | | | | | | | | | |
| N | | | | | | | | | | | | | | | | | | | | | | | | | | |
| O | | | | | | | | | | | | | | | | | | | | | | | | | | |
| P | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Q | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R | | | | | | | | | | | | | | | | | | | | | | | | | | |
| S | | | | | | | | | | | | | | | | | | | | | | | | | | |
| T | | | | | | | | | | | | | | | | | | | | | | | | | | |

| SCHEDULED
△ ESTIMATED COMPLETION
▲ COMPLETED

Attachment "A" to
Monthly Progress Report No. 6
Contract AF04(695)-210

# TECHNIQUES FOR IMPROVING COMPUTER RELIABILITY BY LOGICAL DESIGN (SELF-REORGANIZING MACHINES)

by

IRA TERRIS

## 1.0 Introduction

Within the quest for more reliable digital computers, it is desired to examine machines which possess higher reliability by virtue of their ability to operate after permanent failures have occurred. This ability could be possessed by machines which have the ability to properly evaluate seemingly wrong results (by program methods), by building machines with redundant parts, by machines which have self-reorganizing capability, or by unknown and unspecified methods. As used herein, the expression 'self-reorganizing machines' denotes machines which have a minimum of redundancy and, thus, recover from a failure by, wherever possible, excluding failed equipments from the paths of information flow or replacing them with equipments which had nonessential roles in the machine's organization, rather than replacing the failed equipments with redundant units. Thus, reliability is increased at a reduction or degradation of the machines capability rather than by pure redundancy.

One can envision rather quickly how such schemes can be implemented. The question which arises and which must be answered is, "After implementation of these self-reorganizing schemes, do I have a machine which has a longer mean time between catastrophic failures or has the equipment required to achieve self-reorganization negated the reliability gain?"

Because there are many factors which cannot be expressed in a general mathematical form, it appears that the only way to answer the above question is by an example of a machine design. For this purpose, a rather simple machine was chosen and designed in forms with and without self-reorganizing capability. This design effort yielded a quantitative measure of reliability gain by self-reorganization and a feeling for the areas in which further work should be performed.

## 2.0 Self-Reorganizing Machines

To permit operation after the occurrence of errors or failures, it is necessary to be able to detect, classify (transient or permanent), and recover from errors. The methods that one uses for detection, depend on the time which is allotted for error detection. (This time is a variable depending on the error being caused by the loss in operational calculation, the error state before this failure, etc.) If, when a failure occures, the allowed detection time is sufficiently long, program methods suffice; while if this time is too short, hardware must be included in the machine to perform the error detection function. Since determination of the method of error detection is based on the use of the machine, it is considered beyond the scope of this memorandum.

After detection, it is necessary to classify the error as a transient or permanent failure. Since there is a time dependence in the former case, the obvious method of classification and recovery if the fault was a transient is recomputation. Of course, as in the case of a lost accelerometer input, recomputation is not always possible. In these cases, the time variation may be used for classification (for example, see what happens when next input occurs), but, recovery must be effected by other procedures.

If, however, the error is due to a permanent failure, it is necessary to localize the failure and remove the failed equipment from the path of information flow. The problem of failure localization for the self-organizing machine discussed herein should not be very different from the usual machine. However, rather than enter a discussion of the localization methods for self-reorganizing machines at this time, it is deemed better to study self-reorganizing techniques and obtain an estimate of their effect on machine reliability.

## 3.0 Self-Reorganization Techniques

Machines which have only one organization have that organization specified by the wiring of the machine. If the organization of the machine may be specified by the program, the machine is considered self-reorganizing and the manner in which the reorganizations are specified determines the degree of sophistication of the system. Since this study is concerned with machines of the present or near future, we are limited to rather unsophisticated systems.

Since the machine organization must be continually specified, it is necessary that the information specifying organization be in an active storage element or flip-flop register. This register controls logic which controls the transfers of information from register-to-register, thereby specifying the machine's organization. To change the organization, it is merely necessary to alter the contents of this register by a command (this register looks like a buffer register).

First, let us consider the number of organization states which a machine may assume. It is possible for a word-split memory* and its associated electronics to have as many as 28 failure states and have some operable storage remaining. Thus, including the totally operable state, there are 29 states for the memory. If the arithmetic unit may be split, there are three states for it. The control unit, with its four to six registers and instruction decoder, may have from 16 to 64 states, and the input-output unit, which usually has from 5-15 registers, has at least 32 states. Thus, by multiplying these numbers, it is seen that the entire computer may have in the order of 50,000 states. By proper design, it may be possible to limit the states to 1,000. In this case, the state register would have 10 flip-flops and be comparable in length, and nearly equally prone to failure as any half register. Thus, the state register should have a single error-correcting facility which requires an additional four flip-flops. Now, logic is needed to decode the state register. With a single error-detecting code, each state of the computer is specified by 14 states of the register. Using AND/OR logic, we require 14 thirteen input AND gates and one 14 input OR gate or 196 diodes per machine state. This is an excessive number of components and leads to the conclusion that the state register should be divided. Thus, one or two flip-flops are associated with each computer equipment block and, since one or two flip-flops are much more reliable than a block of equipment, no redundancy of the flip-flops is required.

To further justify the last statement, it is noticed that a failure of the state flip-flop can cause total system failure if, on failure, it enters the state used when there is no failure in its associated equipment and, indeed, there is a failure in the associated equipment. But, this requires two failures and is much less probable. The only single failures of the state indicating equipment which causes total failure are where both outputs are low indicating to the logic to use neither the original equipment nor its replacement, or where both outputs are high indicating use of two equipments for one function.

To obtain a quantitative knowledge of the situation, consider a 12-flip-flop register and its state flip-flop. Let p be the probability of one flip-flop operating correctly. The probability of correct operation is $p^{13}$. The probability of failure of the state flip-flop is q, which is much less than unity. The probability of two failures, one of which is the state flip-flop, is $12qp^{11}q = 12q^2 << q$. Thus, very little is gained by using redundancy of the state flip-flop.

*A Holick, "Word Split Technique as a Means to Increase the Failure Tolerance Level in Core Memories," Hughes Aircraft Co, Report No SRS-622, 15 February 1963.

As suspected, certain machine organizations are more amenable to self-reorganization than others, and self-reorganization techniques used vary with the machine block.

The two self-reorganization procedures which are used in this paper are:

(1) Exclusion of failed portions from the path of information flow, and

(2) Replacement of the failed part.

The components in a machine are used either in logic or storage devices. Generally speaking, the logic is of a nonrepetitive nature, while the storage has a repetitive structure. Further, the ratio of components to input and output signals is higher for storage than for logic. Thus, it appears that the application of self-reorganization techniques to storage devices is more profitable than to logic.

### 3.1 Elimination of Registers

First, let us examine the problem of removing a register from the machine organization. From the logic viewpoint, the register is a holding element and it only affects its outputs as it is a unilateral element. Thus, register removal is accomplished by setting the outputs to a logic state which permits proper operation of the remainder of the machine. Since information flow is unidirectional, the inputs to the register are unaffected by the failed register, and nothing need be done on the input side from a logic viewpoint. (From a circuit viewpoint, this may not be true, but we shall not undertake a failure mode analysis at this time.) However, it may be desirable to prevent these inputs from being lost. In that case, the inputs must be routed to an alternate register. This reorganization (elimination of a register) is accomplished thru the logic associated with the state flip-flop of that register.

For an example of the above, let us consider a failure in the least significant half of the accumulator register and assume that this failuse shall be eliminated by dropping the least significant half of the register. If this accumulator is in a parallel arithmetic unit, one can prevent the least significant half of the accumulator from affecting the most significant half by eliminating the least significant half from the shift chain, bringing the initial carry into the least significant bit of the most significant half in the adder, breaking the carry chain in the adder, causing conditional jump gates to disregard the least significant half, and reducing the number of cycles in multiply and divide by a factor of two. It should be noticed that it is not necessary to cause the register output for the least significant half to be set to zero as the junk in this half register is no better nor worse than zero, or to disable the adder output to the least significant half. The above can be accomplished with seven gates (two for shifting, one for initial carry, one for the conditional jump gates, one to break the carry chain, and two to alter the cycle counter).

If the arithmetic unit is serial rather than parallel, the least significant half of the accumulator is dropped in an analogous manner. In this case, only six elements are needed to alter the cycle and bit counters and to get the register output from the most significant half.

The problem of eliminating the most significant half and using the least significant half for the most significant half is slightly more complicated for a parallel organization but identical to the above for a serial organization. The only difference in the parallel case is the enabling of the logic to permit the most significant bit of the least significant half to be a sign bit.

### 3.2 Replacement of Registers

There exist in a machine certain registers which, by present thinking, are essential to the operation of a digital computer. However, by some future technique, it may be possible to accept failures in these registers or to eliminate these registers in part or whole from the structure and continue to operate successfully. Examples of registers of this type are the memory address register and the instruction register. These registers are generally not merely delays, but inputs to a decoder, and, as such, their content is operated upon in parallel. Thus, replacement of these registers requires gates at the output of each flip-flop (assuming serial input to the register).

It should be noticed that there exists a restriction in register replacement which results from the use of different types of registers in the computer. These register types are:

(1) Shift-in - shift-out with no provision for parallel readout; i.e., core-shift registers,

(2) Counters, and

(3) General flip-flop registers (serial and parallel capability for both inputs and outputs).

Types 2 and 3 suffice for a computer, but where Type 1 can be used, they are more economical. The restrictions on register replacement are Type 1 can be replaced by either Types 1 or 3, and Types 2 and 3 can be replaced by only Types 2 and 3, respectively.

### 3.3 Evaluation of Self-Reorganization Techniques

As mentioned in Section 1.0, it is desired to obtain a quantitative measure of the effect of self-reorganization techniques on machine reliability. Although this measure will not be an upper bound on the usefulness of these techniques, the measure will indicate if these techniques indeed do improve the reliability, or whether the gain is offset by the components needed to implement the techniques, where further improvement is most likely to be achieved, and where further improvement is needed.

Since a measure of the reliability improvement is best achieved by doing a design (as we need component counts), we do that in the following section.

## 4.0 Self-Reorganizing Computer for Inertial Guidance System

The computer which is used in this example is shown in its standard (not self-reorganizing) form in Figure 1. This computer accepts acceleration and time pulses, and outputs command signals and gyro torquing pulses.

### 4.1 Standard Computer

The following machine organization is assumed:

(1) Random access core memory of 4,096 words with parallel readout and 4 microsecond cycle time.

(2) Serial arithmetic unit operating at a 2 megacycle per second rate.

(3) True (two's) complement arithmetic.

(4) Orders are 15 bits long (2 orders/word).

(5) Numbers are 30 bits long.

(6) One index register.

(7) Integrated circuits are used wherever possible.

(8) Basic clock rate is 2 mc.

Further details, such as instruction repertoire and order code are given in the appendix.

### 4.1.1 Control Unit

The control unit is rather standard in construction and, therefore, covered very briefly.

The instruction register is 7 flip-flops and accepts information in a serial manner for decoding by the instruction decoder.

The index register holds a 6-bit number (0-63) for adding to the operand address when desired.

The program counter (12FF) holds the address of the next instruction.

The shift counter (5FF) is used during shift, multiply, and divide operation.

The timing counter (8FF) counts bit times and word times on short instructions and during instruction access.

### 4.1.2 Arithmetic Unit

This unit has a serial adder. The memory register is used as the third register.

### 4.1.3 Memory

The memory is a standard random access magnetic core memory of 4,092 30-bit words.

### 4.1.4 Input-Output Unit

This unit has two accumulators, one for accelerometer and time pulses and the other for gyro torquing pulses. Each accumulator consists of a 30 flip-flop shift register and an adder. The pulse inputs are assumed synchronized to the computer by providing the platform coupler with the necessary timing signals.

A 6 flip-flop gyro torque control register tells the platform coupler which torque directions are required for the three axes.

The buffer register has a serial input and output for operation with other equipments and displays.

## 4.2 Self-Reorganizing Computer

The self-reorganizing computer with the same capability as the standard computer, is shown in block diagram form in Figure 2. The features 1-8 listed for the standard computer hold for the self-reorganizing computer. The self-reorganizing capabilities proposed for this example are discussed below. The capabilities described are not to be construed as the best, but only indicative of what can be accomplished.

### 4.2.1 Control Unit

The control unit contains equipment which is generally essential to the operation of the machine and with present techniques no failures can be tolerated. Thus, these equipments must be replaced in case of failure. For this purpose, it is assumed that there is a separate spare counter which may replace any of the other counters in the control unit. Here, a spare is required as no nonessential counter exists. The three registers, memory address, instruction, and index, are the same type as in the arithmetic unit, and it is assumed that one of the four arithmetic unit half registers may replace these registers. The details of replacement are wired into the logic. Allowance is made for operation with failure in any one half register.

The instruction decoder is logic and following the discussion of Section 3.0, is left as an essential equipment.

### 4.2.2 Arithmetic Unit

The arithmetic unit is serial with provision for splitting in half. When the arithmetic unit is split, the bit counter input is shifted left one bit as the word length is cut in half. Also, if the least significant half is used, the new sign flip-flop must have some logic activated. The serial adder is assumed nonredundant for this first pass.

### 4.2.3 Input-Output Unit

This unit has accumulators for accelerometer pulses, gyro torquing pulses, and time. As discussed in Section 4.1.4, there are two shift registers, each with an adder. If a failure in one of these units occurs, the other assumes the entire load by shortening the storage capacity for each quantity. Of course, this means that the computer is required to do more input-output processing and may be required to drop some of its nonessential functions.

The gyro torque control register is replaced by the buffer register on failure. The buffer register, which at time is used infrequently, is replaced by one of the arithmetic unit half registers.

### 4.2.4 Memory

The memory utilizing the word split technique is discussed in detail in the previously referenced report and, therefore, is not discussed in this report.

### 4.3 Computer Reliability

In the above computer organization, a computer whose only essential parts are the serial adder in the arithmetic unit, the instruction decoder in the control unit, the information transfer gates is shown, and the self-reorganizing elements. It remains to determine the reliability of this computer in order to obtain a quantitative measure of the reliability gain. For this purpose, it was necessary to do detailed logic designs on a machine without reorganization capability and one of the same computational capability, but with the above self-reorganizing capability. For the details of these designs, the reader is referred to the Appendix.

From the Appendix, it is seen that the original machine has 771 Micrologic elements excluding memory cores, decoding electronics, and sense electronics. When the self-reorganizing capability is added, the resulting essential equipment is the adder, logic, and reorganizing logic which total 337 elements.

For small probability of failure and assuming that all the Micrologic elements have an equal probability of failure, the probability of catastrophic failure $P_f \cong Nf_rt$. Also, this assumes that only failures in essential equipment contribute significantly to the probability of failure. To a first approximation, this assumption is justified as a catastrophic failure from nonessential equipment requires at least two failures. Thus, we see that the improvement in MTBF is given by the ratio of essential components in the original machine to the self-reorganizing machine.

Thus, in the first cut presented herein, we have achieved an improvement in MTBF of 2.3 ($\frac{771}{337} = 2.3$).

## 5.0 Conclusions

On this first cut at an application, it has been shown that the MTBF of a guidance computer (excluding memory) can be improved by a factor of 2.3. (The improving of MTBF in the memory has been investigated in the previously referenced report, and even greater improvements have been made in that area.)

The example shown in this memo demonstrates that a simple self-reorganizing system has a higher MTBF than its standard counterpart (the gains are not cancelled by the added components), and shows that work along these lines should be continued. Although the computer presented is rather limited in capability, the MTBF improvement is a ratio which will be fairly independent of machine size.

At this time, it should be emphasized that the eliminating or dropping of registers from a machine's organization results in a loss in computational capability. This loss results in less accurate computations, if one chooses to use the half length operands, at no loss in time, or a much slower machine if one requires double precision arithmetic performed by subroutine to compensate for the half length arithmetic unit. Mr A Holick is studying the errors due to degraded mode operation (half-length operands) on midcourse guidance trajectory determination.

## 6.0 Future Work

Herein have been shown the application of two self-reorganization techniques, and from the design presented, one obtains the feeling that with these techniques an upperbound on MTBF improvement is about 400%.

The future work should investigate the finding of other techniques and the detailed application of these techniques to various machine organizations (parallel as well as serial machines).

In particular, future studies should include:

(1) Order codes which result in higher machine reliability.

(2) Program compensation of failed machines (methods for tolerating failures which cause a machine to perform orders in a slightly incorrect manner, or doing failed orders by subroutine).

(3) Study of input-output equipment with interrupt features and its effect on self-reorganizing machines.

I. Terris

I TERRIS

10 May 1963

## APPENDIX

Presented herein are the details for the standard and self-reorganizing machines. These details are needed so that an accurate reliability estimate in which one could display confidence could be obtained. It should be emphasized that, while complete accuracy in component counts would be best, for the purposes of this paper, only relative numbers are required. Thus, any omissions will probably be common to the two machines and the effect will be reduced. Further, a failure mode analysis is required to obtain a true reliability figure, particularly in the self-reorganizing equipment where only certain failures are catastrophic.

The general features of the machine are discussed in Section 4.1. In addition, there is a group select feature for memory addressing which permits instructions to be only 15 bits long. The memory is divided into 16 groups of 256 words each. On "arithmetic" type instructions which have only an 8-bit address, the group is specified by the group from which the order was taken if the group select bit is 1; if it is 0, the group select for the operand is 0000.

### Instruction Repertoire

There are three classes of instructions which are described in detail below.

The jump instructions contain a full 12-bit address which is not alterable by group select or indexing. On jump, the program counter is put in the memory at location 0000 for a return jump. Jumps are always to instructions in the left half of a word.

The arithmetic instructions have an 8-bit address which is indexable. The four most significant bits are specified by a group select feature. If the group select bit (bit 6) is zero, the group select for the operand is zero. If bit 6 is one, then the group from which the instruction was obtained is used. The index register's contents are added to the address on bit 7 equal to one. The index register holds a 6-bit number whose magnitude is in the range 0-63.

The third class of instructions does not require a memory cycle for their execution.

The instruction repertoire follows on the next page.

10 May 1963

UCJ - Unconditional Jump.

JLZ - Jump on contents of A register less than zero.

JCZ - Jump on contents of A register greater than zero.

JOI - Jump on Indicator. The indicator is set on overflow or cycling of index register (see IXR instruction). The indicator is reset on the instruction after it is set.

CAD - Clear Accumulator and Add contents of location specified to the contents of A (C(A)).

ADD - Add contents of A to contents of specified location and put result in A.

SUB - Subtract contents of specified location from C(A) and put result in A.

MPY - Multiply C(A) by C (specified address), and put result in A and Q.

DIV - Divide A (A and Q) by C (specified address) and put result in Q.

EXT - Form the logical product of C(A) and C (specified address) and put result in A.

STR - Store C(A) in specified address. C(A) is not disturbed.

SXR - Set index register to number specified by bits 10-15.

SHL - Shift left by number of places specified by bits 11-15. Bit 6 specifies long or short shift.

SHR - Shift right; bit 6 specifies long or short.

IPT - Input operation which transfers contents of specified interface register (bits 10-15) to A register (30 bits).

OPT - Transfers contents of A to specified interface register.

RGS - Set group select flip-flops (4 most significant bits of program counter) to number specified by bits 12-15 of order.

IXR - Increment the content of the index register and set the indicator flip-flop, if the old content is equal to the number specified by bits 10-15.

SSR - Set state register flip-flop specified by bits 10-14 to the state specified by bit 15*. (This instruction is not implemented in the standard machine.)

The order code for this instruction repertoire is shown in Table 1, attached.

## Timing Sequence

The following indicates the machine sequences during the execution and obtaining of the various instructions. W is used to abbreviate word, and B for bit time. The memory cycle is 4 microseconds or 8 bit times.

W1 B1-B8 Read and rewrite memory for instruction and increment program counter.

W1 B9-B24 Transfer instruction to instruction register (bits 1-7) and memory address register (bits 8-15).

W1 B25-B30 If Type 3 instruction, transfer bits 1-6 to the shift counter.

W2 B1-B30 Execute Type 1 and 3 instructions, all of which require only one word time.

W2 B1-B12 Do address modification of Type 2 instructions.

If STR operation:

W2 B13-B16 Read memory (clear).

W2 B17-W3 B16 Transfer C(A) to memory register.

W3 B17-B20 Write memory.

If not STR operation:

W2 B13-B20 Obtain operand from memory.

*Note that the register as a whole cannot be treated as one shift register to save the addressing logic as one failed flip-flop which must be tolerated would not allow shifting thru it.

| | |
|---|---|
| W2 B1-B30 | If MPY, put multiplier in Q and clear A. Set shift counter to 30 on MPY or DIV. |
| W3 | Execute until shift counter is stepped to zero if MPY or DIV, or for 1 word time. |
| W2 B19-B30 | If Type 1 or 3 instructions, transfer PC to MAR. |
| W3 B19-B30 | If Type 2 instruction, transfer PC to MAR. |

## Detailed Block Diagram of Standard Computer

A detailed block diagram of the standard computer is shown in Figures 3-6. Here, are shown the paths of information flow and the flip-flop designations for each register. Associated (but not shown) with each register is a gated clock pulse amplifier for permitting information transfers at the proper time. Putting the gating on the clock pulses minimizes the logic.

## Component Count

To obtain an exact component count, the logic should be implemented using the logic function blocks available (NOR or NAND logic). However, to reduce the labor considerably, the equations were written in AND-OR form, which yields almost the same count as NAND or NOR form. Where counters and registers are used, the exact component counts were obtained from the configurations given by Fairchild for their Micrologic.

## Registers and Counters

The following element counts include two elements per shift register stage, three elements per shift register stage where parallel entry is also permitted, three elements per counter stage, and four elements per counter stage where shift entry also is required. The estimates also include any adders or buffer elements for fan-out or interface required.

### Input-Output Unit (Figure 3)

| | | |
|---|---|---|
| ATR | - | 76 elements |
| GTR | - | 76 elements |
| GTC | - | 18 elements |
| BR | - | 60 elements |

Memory Unit (Figure 4)

| | | |
|---|---|---|
| MR | - | 90 elements |

Arithmetic Unit (Figure 5)

| | | |
|---|---|---|
| AR | - | 60 elements |
| QR | - | 60 elements |

Control Unit (Figure 6)

| | | |
|---|---|---|
| MAR | - | 24 elements |
| IR | - | 14 elements |
| XR | - | 12 elements |
| B&WC | - | 21 elements |
| SC | - | 15 elements |
| PC | - | 36 elements |

## Logic

Equations were written to obtain the counts shown below in AND-OR form, but are not included in this memo. By very simple transformations, these equations are converted to NOR or NAND logic. Then, using the fan-out and fan-in realizable for the Micrologic elements, counts were obtained.

Decoders

| | | |
|---|---|---|
| Instruction decoder | - | 29 elements |
| Bit and Word-time decoder | - | 34 elements |
| Shift counter decoder | - | 10 elements |

Information Gating

This gating is used at the register input or on the clock pulse amplifier.

## Input-Output Unit

| | | |
|---|---|---|
| GTC | - | 2 elements |
| GTR | - | 5 elements |
| BR | - | 2 elements |

## Memory Unit

| | | |
|---|---|---|
| Read Memory (1/2 cycle) | - | 2 elements |
| Write Memory (1/2 cycle) | - | 1 element |
| Read and Write | - | 15 elements |
| MR | - | 3 elements |

## Control Unit

| | | |
|---|---|---|
| MAR | - | 9 elements |
| IR | - | 3 elements |
| XR | - | 2 elements |
| B&WC Reset | - | 7 elements |
| SC | - | 4 elements |
| PC | - | 6 elements |

## Arithmetic Unit

| | | |
|---|---|---|
| Q | - | 5 elements |
| A | - | 12 elements |
| Adder input 1 | - | 5 elements |
| Adder input 2 | - | 4 elements |
| Adder including overflow FF, carry FF, Add-Subtract control, sign corrections, initial carry, etc | - | 40 elements |

10 May 1963

From the above, it is determined that 571 elements are needed for registers and 200 elements for logic and gating.

Self-Reorganizing Machine

The self-reorganizing machine has the same basic structure as the original machine, but has the self-reorganizing capability mentioned in the body of this report.

This capability is shown in detail in Figures 7-10. Using Micrologic, it is determined by a count that 17 flip-flops and 120 logic elements for a total of 137 elements are needed for the self-reorganizing capability.

The decoding of state register flip-flops 22 and 23 is as follows:

| | | |
|---|---|---|
| (22')(23') | - | Full length arithmetic |
| (22)(23') | - | Use MSH registers only |
| (22')(23) | - | Use LSH registers only |

State register flip=flops 1 and 2 are used for finding the alternate register (ALT) for the control unit.

ALT = (Q LSH)(22)(2')•+ (Q MSH)(22)(2) + (A LSH)(22)(2) + (A MSH)(23)(1) [redacted] 10 May 1963

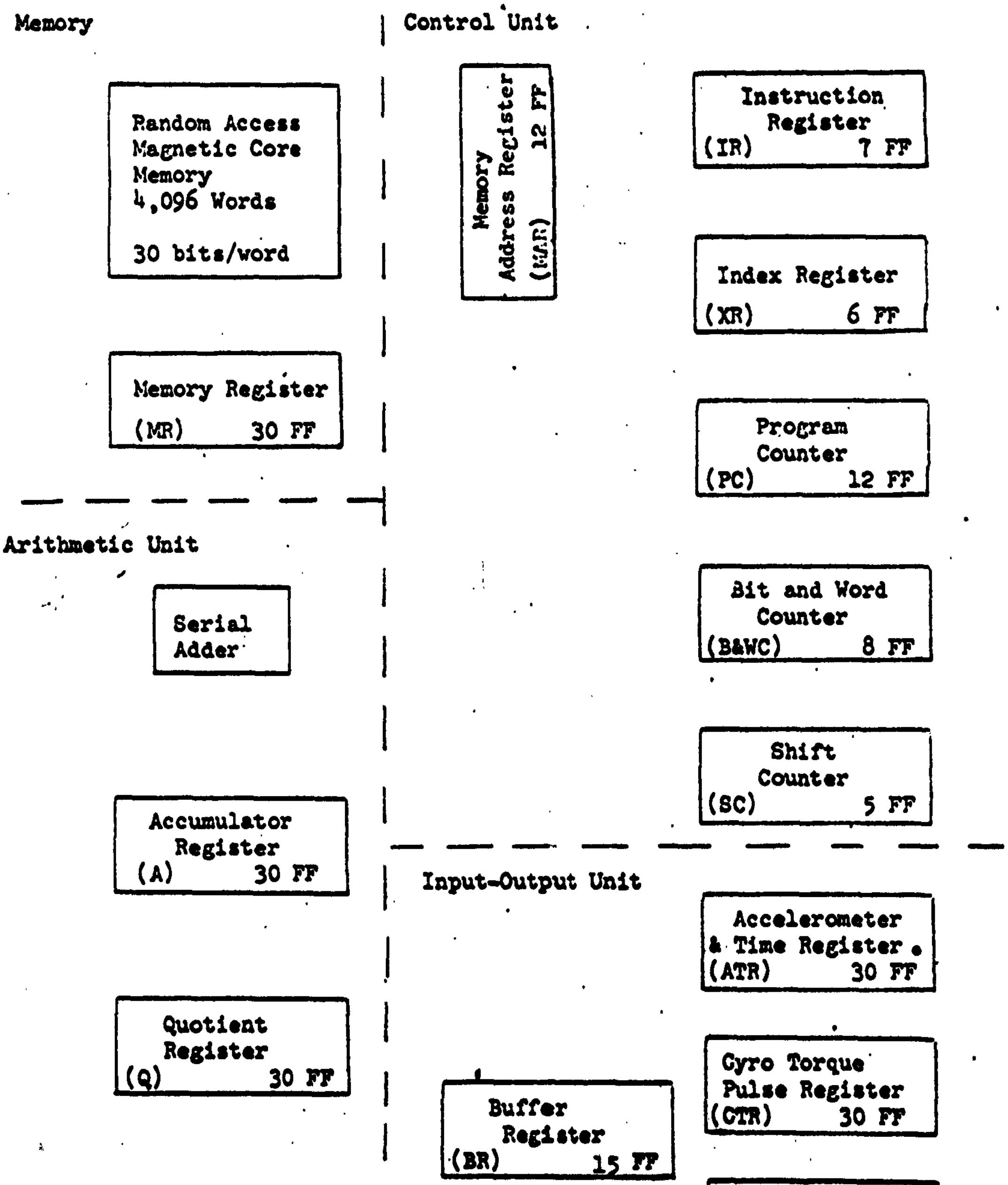
BIT POSITIONS

| ORDER | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UCJ | 0 | 0 | 0 | | | | | | | | | | | | |
| JLZ | 0 | 0 | 1 | | | | | | | | | | | | |
| JOI | 0 | 1 | 0 | | | | | | | | | | | | |
| JPN | 0 | 0 | 1 | | | | | | | | | | | | |
| CAD | 1 | 0 | 0 | 0 | 0 | GS | XR | | | | | | | | |
| ADD | 1 | 0 | 0 | 0 | 1 | GS | XR | | | | | | | | |
| SUB | 1 | 0 | 0 | 1 | 1 | GS | XR | | | | | | | | |
| MPY | 1 | 0 | 1 | 0 | 1 | GS | XR | | | | | | | | |
| DIV | 1 | 0 | 1 | 1 | 1 | GS | XR | | | | | | | | |
| EXT | 1 | 0 | 1 | 0 | 0 | GS | XR | | | | | | | | |
| STR | 1 | 0 | 0 | 1 | 0 | GS | XR | | | | | | | | |
| SXR | 1 | 1 | 0 | 0 | 0 | 1 | 0 | d | d | | | | | | |
| SHL | 1 | 1 | 1 | 0 | 0 | L/S | d | d | d | d | | | | | |
| SHR | 1 | 1 | 1 | 0 | 1 | L/S | d | d | d | d | | | | | |
| IPT | 1 | 1 | 1 | 1 | 0 | d | d | d | d | | | | | | |
| OPT | 1 | 1 | 1 | 1 | 1 | d | d | d | d | | | | | | |
| RGS | 1 | 1 | 0 | 0 | 0 | 0 | 1 | d | d | d | d | | | | |
| IXR | 1 | 1 | 0 | 0 | 0 | 0 | 0 | d | d | | | | | | |
| SSR | 1 | 1 | 0 | 1 | d | d | d | d | d | | | | | | 0/1 |

d - don't care

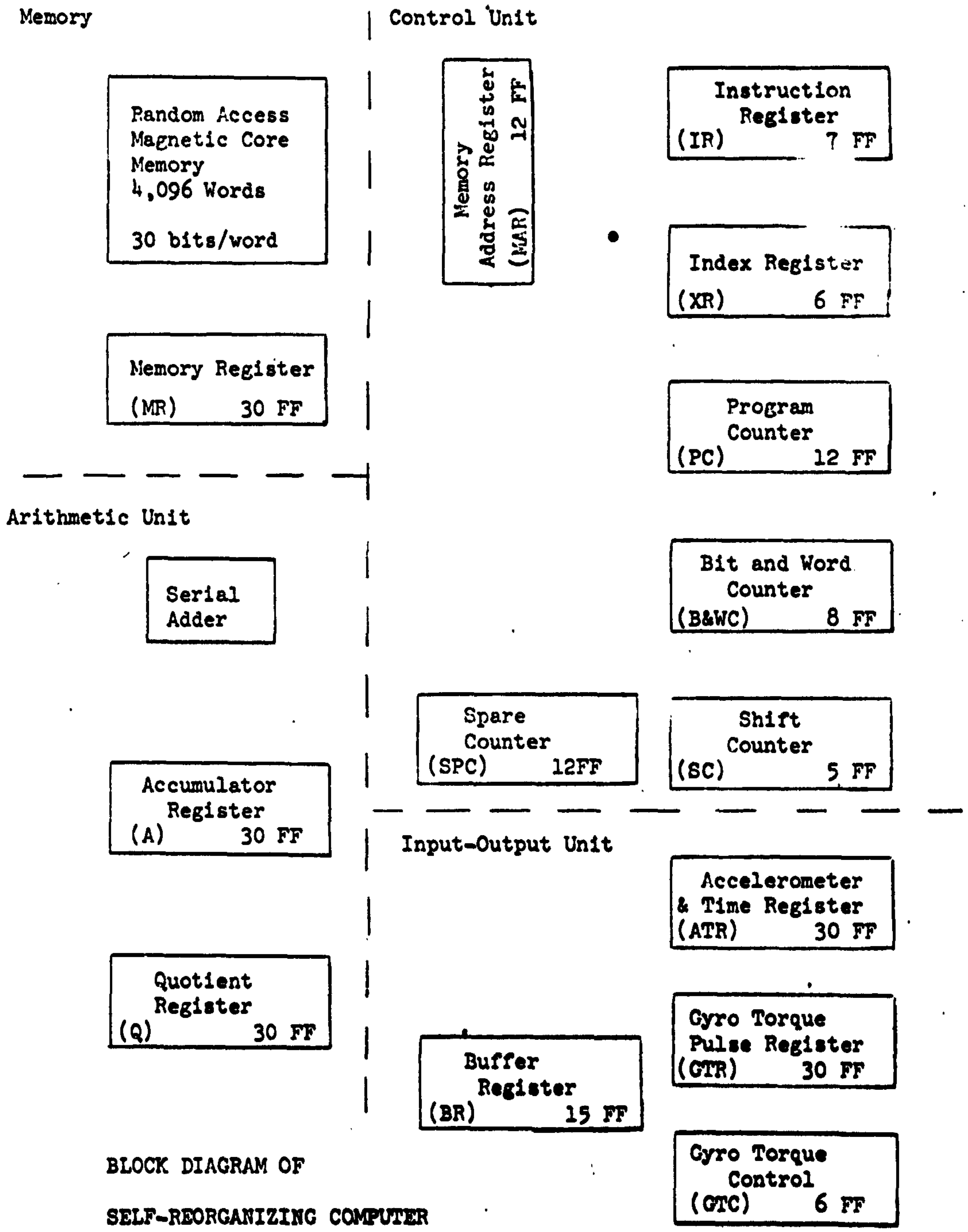
Blanks are address bits

ORDER CODE - TABLE 1



BLOCK DIAGRAM OF STANDARD COMPUTER

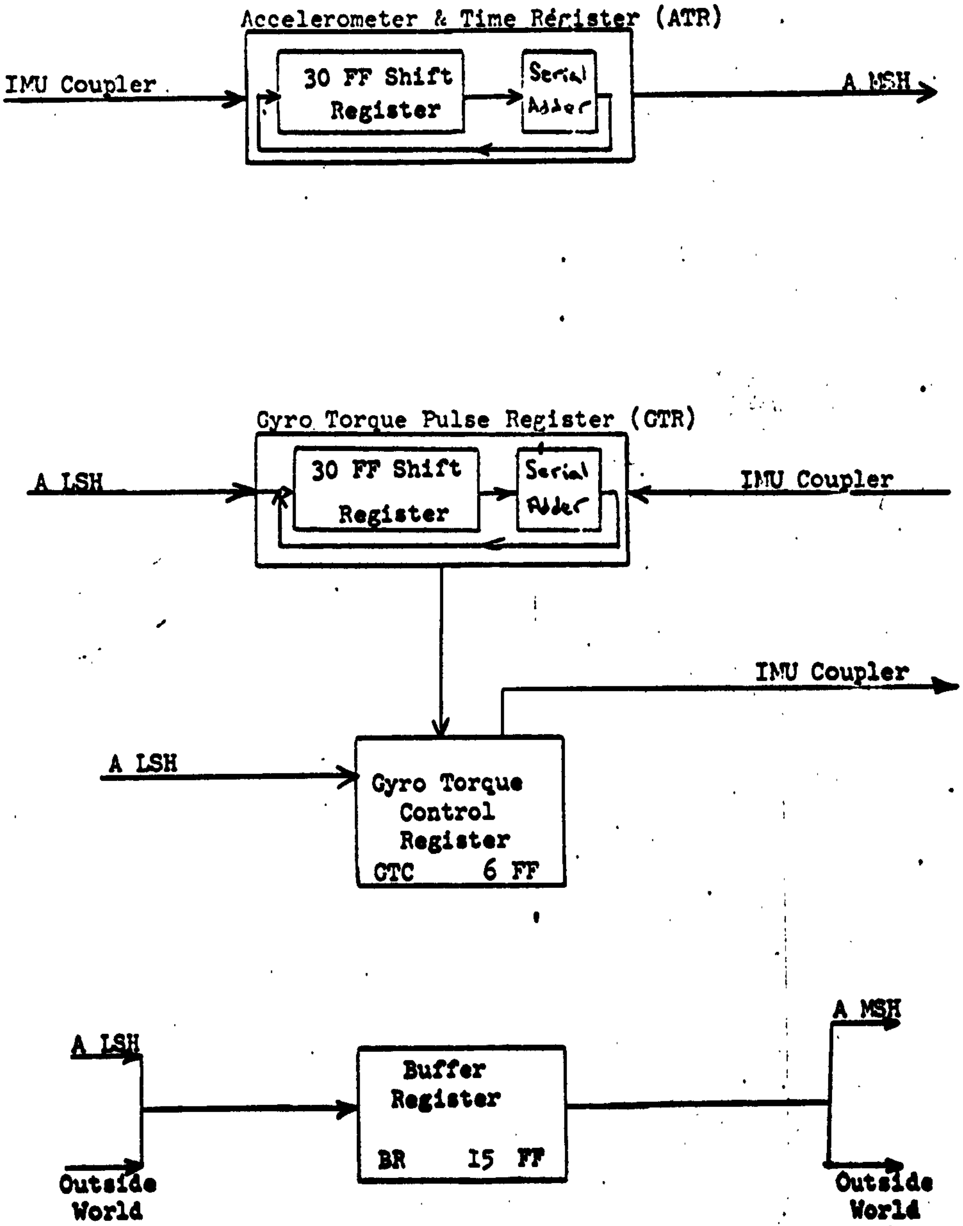
FIGURE 1


Memory
Control Unit
Random Access Magnetic Core Memory 4,096 Words 30 bits/word
Memory Address Register (MAR) 12 FF
Instruction Register (IR) 7 FF
Index Register (XR) 6 FF
Memory Register (MR) 30 FF
Program Counter (PC) 12 FF
Arithmetic Unit
Serial Adder
Bit and Word Counter (B&WC) 8 FF
Spare Counter (SPC) 12FF
Shift Counter (SC) 5 FF
Accumulator Register (A) 30 FF
Input-Output Unit
Accelerometer & Time Register (ATR) 30 FF
Quotient Register (Q) 30 FF
Gyro Torque Pulse Register (GTR) 30 FF
Buffer Register (BR) 15 FF
Gyro Torque Control (GTC) 6 FF

BLOCK DIAGRAM OF

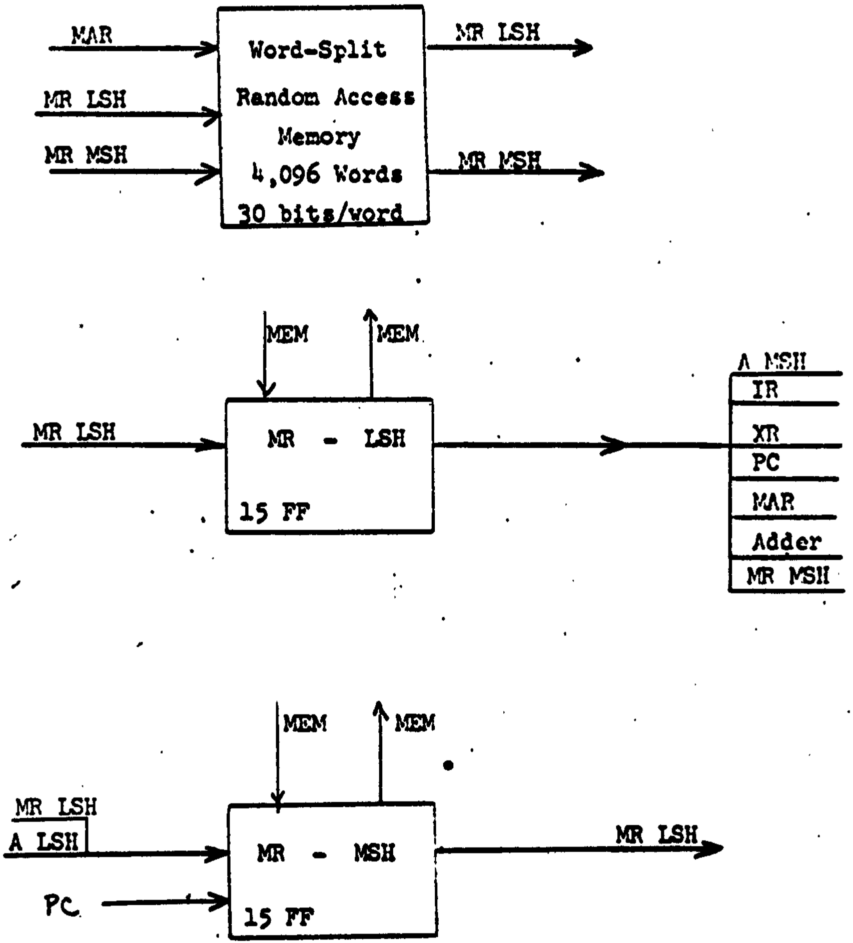
SELF-REORGANIZING COMPUTER

FIGURE 2



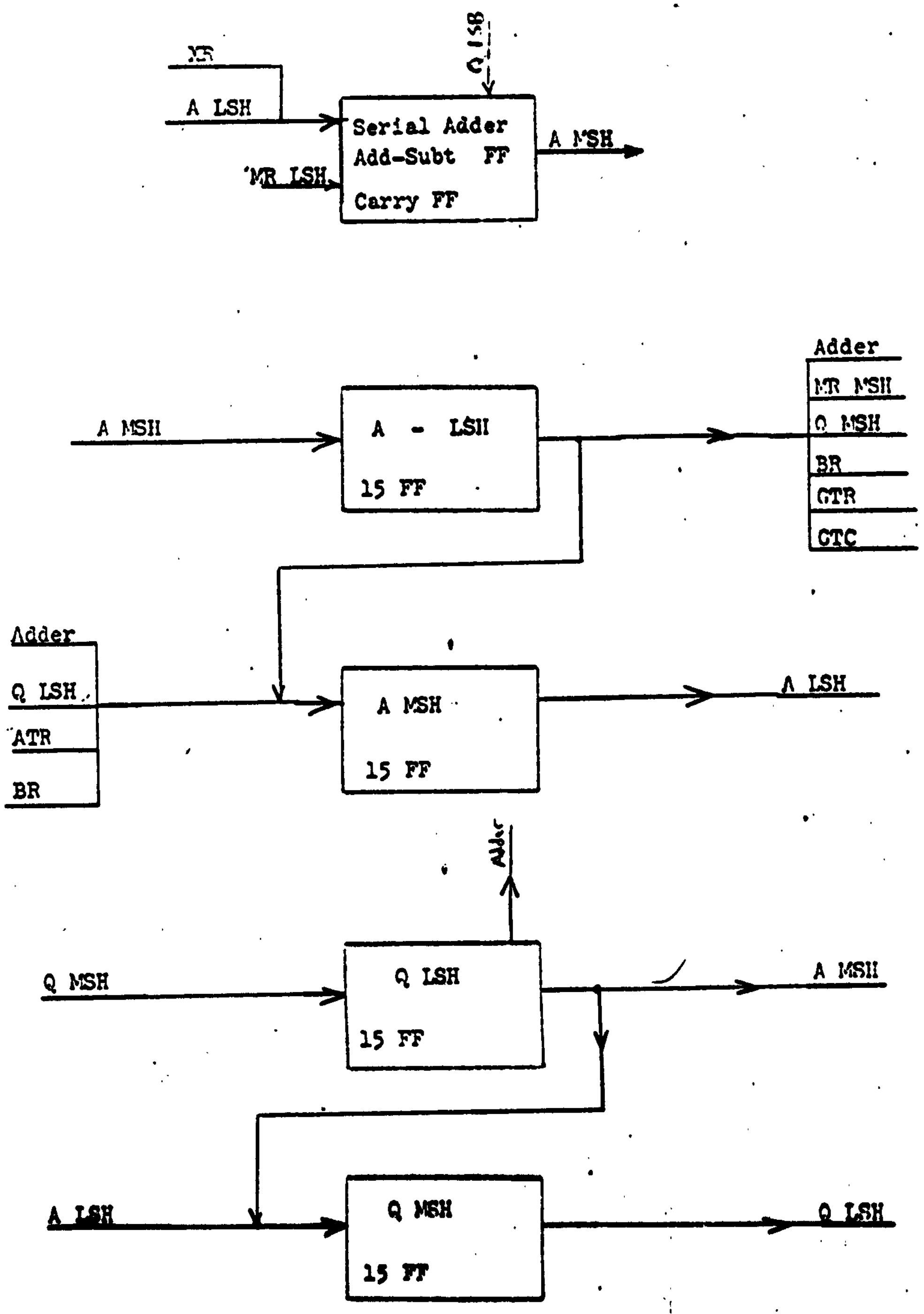
INPUT - OUTPUT UNIT

FIGURE 3



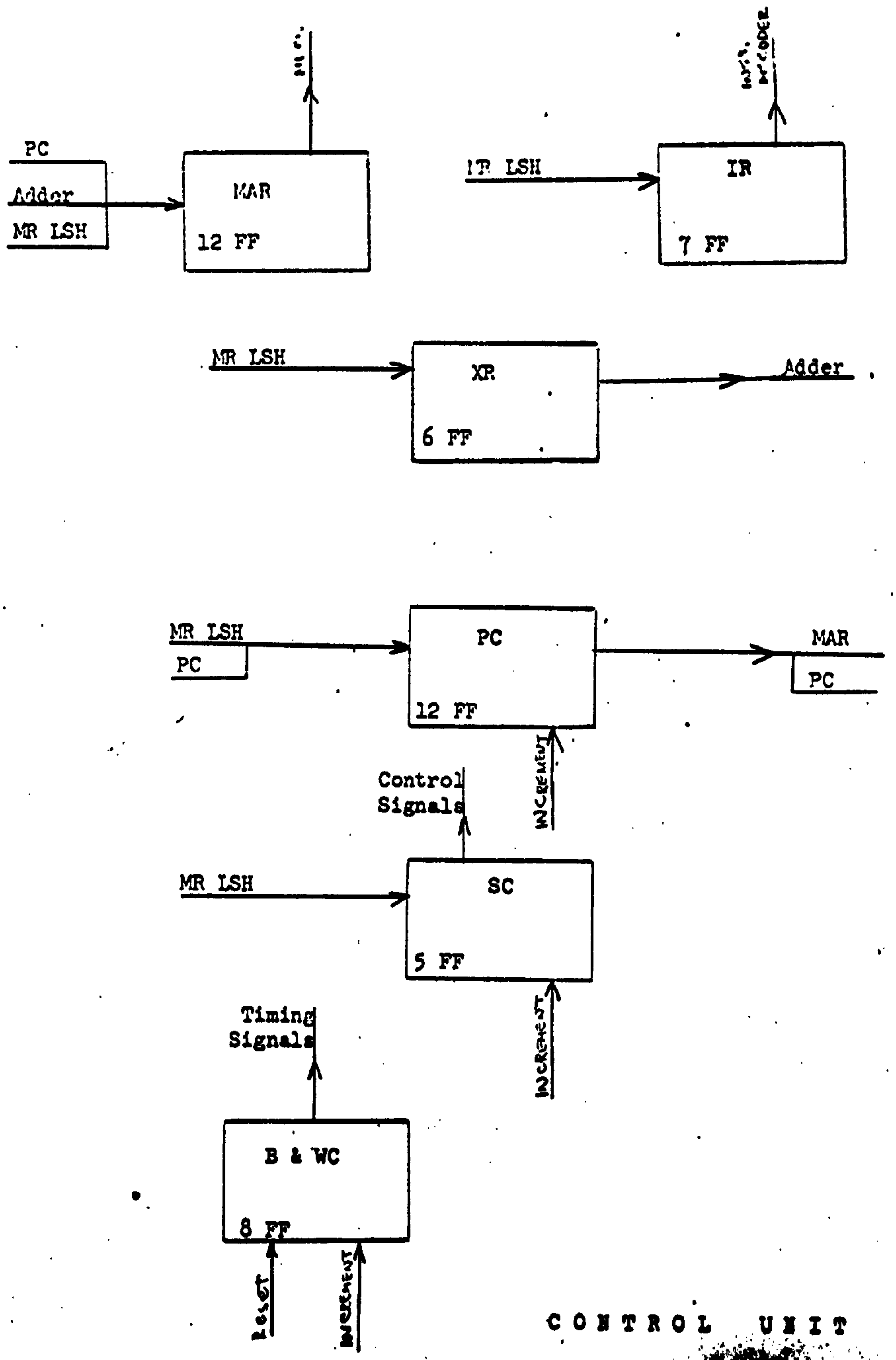
MEMORY UNIT

FIGURE 4



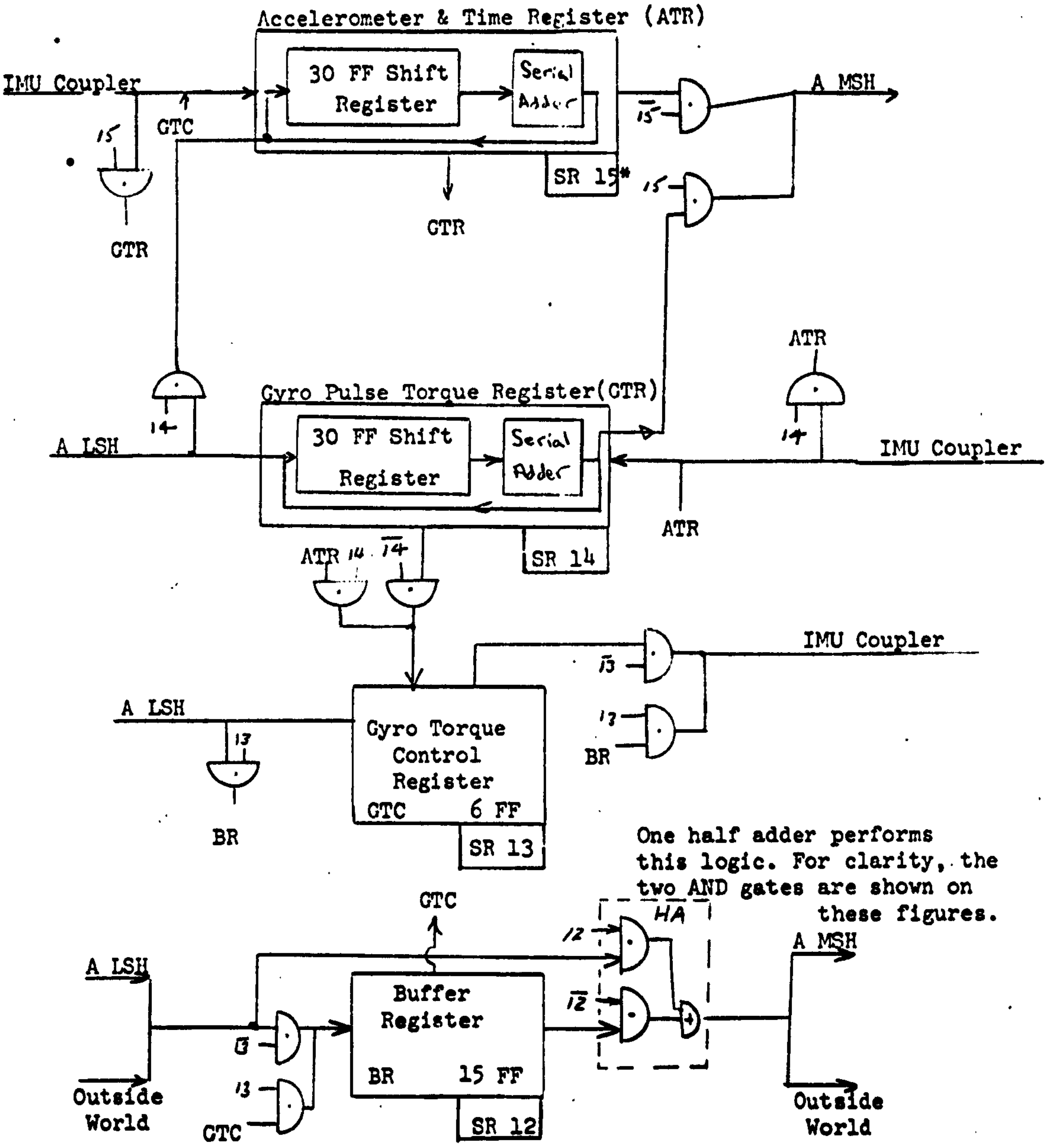
ARITHMETIC UNIT

FIGURE 5



CONTROL UNIT

FIGURE 6

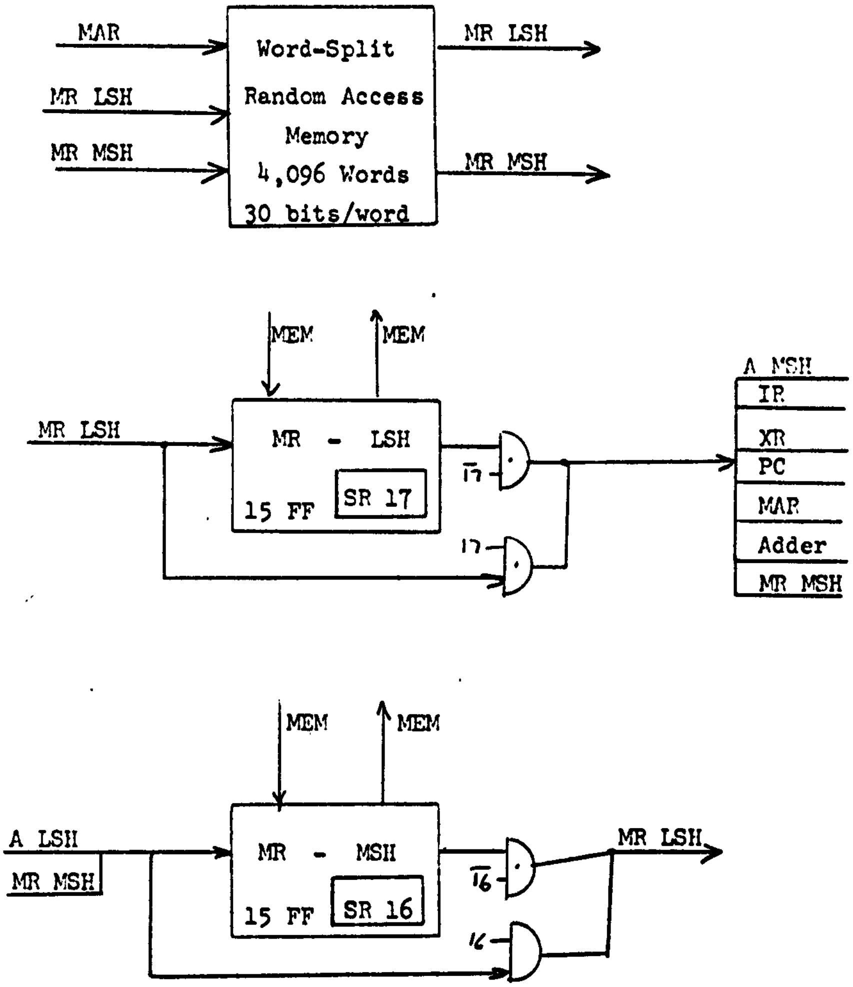


One half adder performs this logic. For clarity, the two AND gates are shown on these figures.

INPUT - OUTPUT UNIT

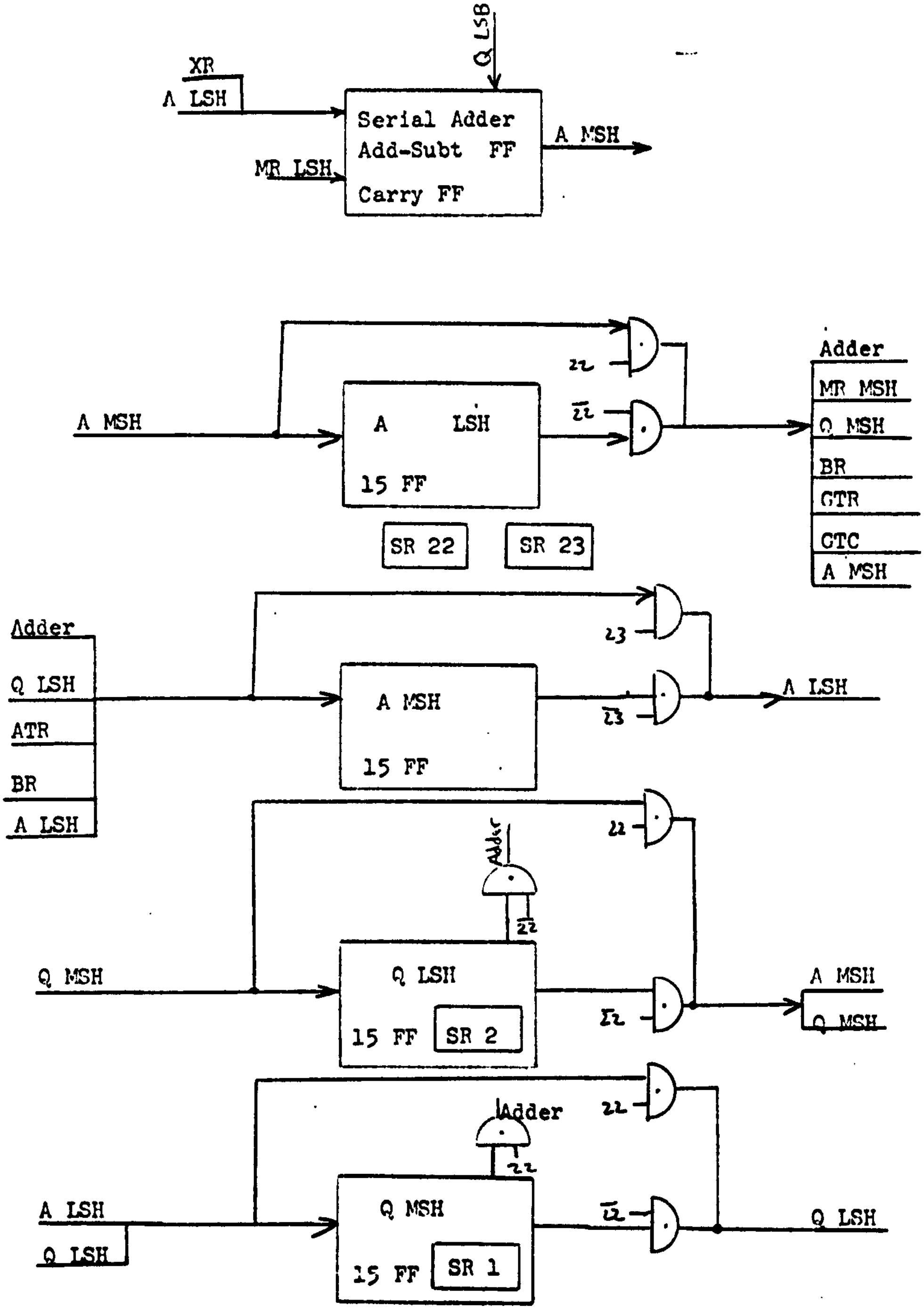
FIGURE 7

* This is FF15 of the state register.



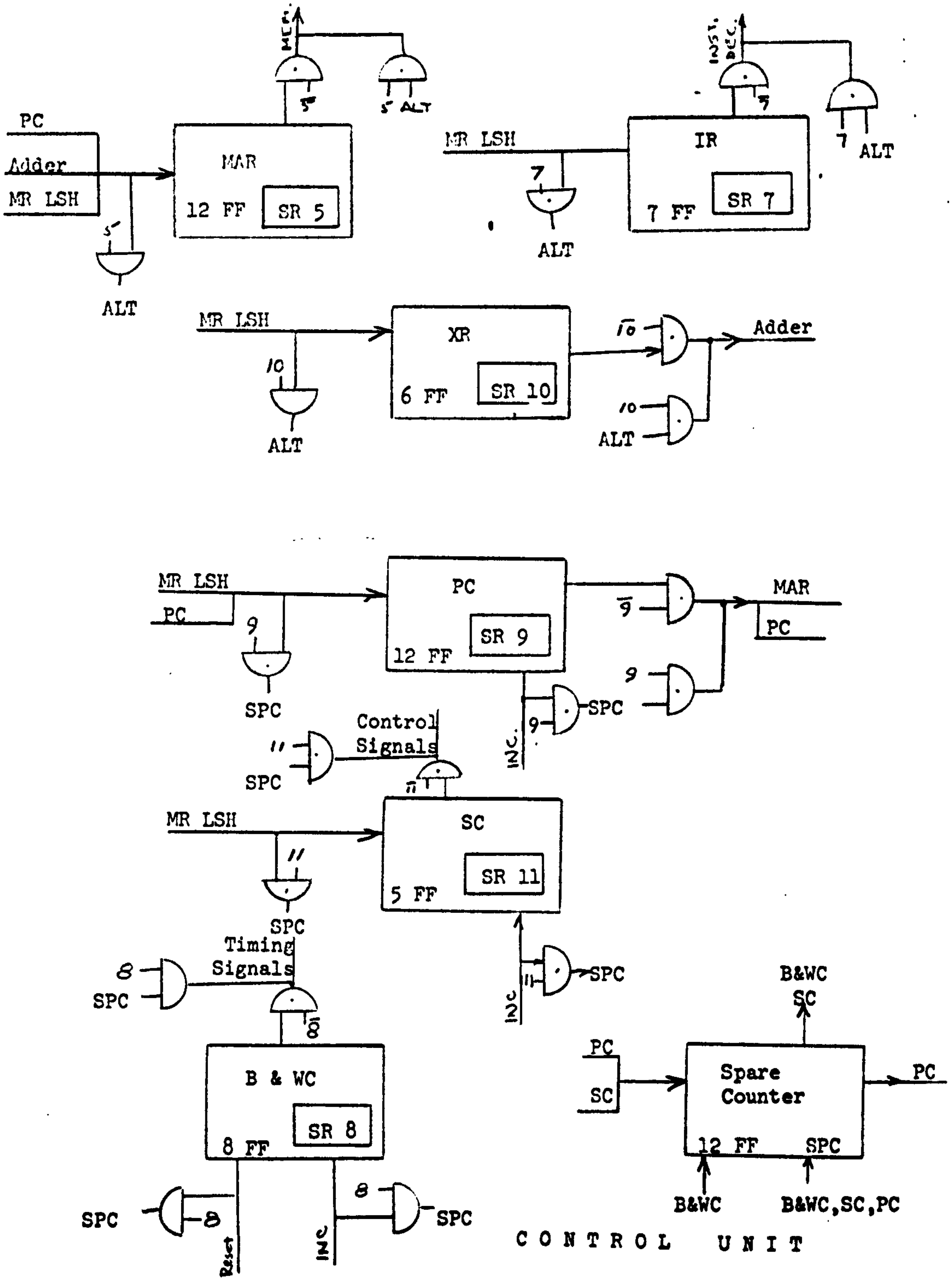
MEMORY UNIT

FIGURE 8



ARITHMETIC UNIT

FIGURE 9



CONTROL UNIT

FIGURE 10